Bosna i Hercegovina
Federacija Bosne i Hercegovine
SREDNJOBOSANSKI KANTON
Ministarstvo obrazovanja, nauke, mladih, kulture i sporta Srednjobosanskog kantona/Ministarstvo obrazovanja, znanosti, mladih, kulture i športa Kantona Središnja Bosna

KURIKULUM NASTAVNOG PREDMETA
PROGRAMIRANJE

ZA IT GIMNAZIJU

Travnik, maj 2025.

Bosna i Hercegovina
Federacija Bosne i Hercegovine
SREDNJOBOSANSKI KANTON
Ministarstvo obrazovanja, nauke, mladih, kulture i sporta Srednjobosanskog kantona/Ministarstvo obrazovanja, znanosti, mladih, kulture i športa Kantona Središnja Bosna

KURIKULUM NASTAVNOG PREDMETA

PROGRAMIRANJE

ZA IT GIMNAZIJU

Travnik, maj 2025.

Kurikulum nastavnog predmeta
Programiranje

Izdavač: Ministarstvo obrazovanja, nauke, mladih, kulture i sporta Srednjobosanskog kantona/Ministarstvo obrazovanja, znanosti, mladih, kulture i športa Kantona Središnja Bosna

Za izdavača: Bojan Domić, ministar

Stručni tim za razvijanje, prilagođavanje i inoviranje predmetnih kurikuluma i njihovu primjenu u osnovnim i srednjim školama na području Srednjobosanskog kantona u kojima se nastavni proces realizira na bosanskom jeziku:
Nezira Fuško, voditeljica Stručnog tima
doc.dr.sc Nešad Krnjić, voditelj radne skupine
Amra Mirojević, MA., član
Arnela Šabanović, MA., član

Recenzenti:
prof.dr.sc (SBK)
prof.dr.sc (SBK)

Tehnička priprema i uređenje:
Ministarstvo obrazovanja, nauke, mladih, kulture i sporta Srednjobosanskog kantona/Ministarstvo obrazovanja, znanosti, mladih, kulture i športa Kantona Središnja Bosna

SADRŽAJ

A/ OPIS PREDMETA

U savremenom digitalnom dobu, sve veća prisutnost računara, pametnih uređaja i tehnologija s integrisanim inteligentnim modulima stvara potrebu za aktivnim razumijevanjem i korištenjem programiranja, ne samo kao korisnika, već i kao kreatora. Programiranje se danas nameće kao neizostavna kompetencija koja omogućava učenicima da ovladaju temeljnim informatičkim konceptima kao što su algoritmi, logika i matematičko razmišljanje, čime se otvaraju vrata ka rješavanju problema iz različitih oblasti nauke, industrije i svakodnevnog života.

Kroz proces izučavanja programiranja, učenici ne samo da razvijaju tehničke vještine, već i sposobnosti analitičkog razmišljanja, preciznosti, kreativnosti i samopouzdanja. Upoznaju se s raznim programskim jezicima i njihovim specifičnostima, ali istovremeno stiču uvid u univerzalne principe strukturnog i objektnog programiranja. Bez obzira na izbor jezika, učenici uče razlikovati i razumjeti različite programske paradigme, razvijati kritički stav prema alatima koje koriste, te pravilno primjenjivati koncepte kao što su sekvenca, selekcija i iteracija. Kroz kreiranje vlastitih aplikacija učenici jasno uočavaju razliku između računarstva i informatike, te se osposobljavaju za uspješno snalaženje u savremenom, dinamičnom tehnološkom okruženju.

Učenje programiranja kod učenika treba da se odvija postupno, kroz primjenu raznovrsnih programerskih tehnika i ovladavanje savremenim konceptima algoritmizacije rješenja problema. Poseban akcenat stavlja se na prevođenje algoritama u konkretan programski jezik, čime se kod učenika razvijaju i tehničke i intelektualne vještine. Predmet Programiranje aktivno promoviše timski rad – kroz parove, manje grupe i veće projektne timove – što naročito dolazi do izražaja u višim razredima. Takav pristup podstiče komunikaciju, saradnju i sposobnost zajedničkog pronalaženja rješenja.

Učenici kroz praktične zadatke i rad na projektima usvajaju napredne koncepte odabrane programske paradigme, pri čemu razvijaju istraživački duh, takmičarski entuzijazam i kritičko-analitički stav prema programerskim tehnikama. Rad sa stvarnim ulaznim podacima i rješavanje konkretnih problema dodatno osnažuju njihovu sposobnost prilagođavanja kodu, testiranja i prihvatanja izmjena.

Predmet Programiranje izučava se kontinuirano tokom sve četiri godine IT gimnazijskog obrazovanja, od prvog do četvrtog razreda, koristeći jedan ili više programskih jezika koji se temelje na objektno-orijentisanoj paradigmi. Nastava obuhvata širok spektar programerskih filozofija i njihovu praktičnu primjenu, pri čemu učenici stiču znanja kako iz tehničkog, tako i iz algoritamskog pristupa rješavanju problema. Ovaj predmet ne funkcioniše izolovano – on integrira i proširuje znanja iz oblasti informacionih i komunikacionih tehnologija, nudeći učenicima dublje razumijevanje osnovnih principa rada računara, načina obrade podataka i interpretacije programskih komandi.

Programiranje se oslanja na algoritamski način razmišljanja i duboko je povezano s matematičkim i logičkim vještinama stečenim tokom obrazovanja. Dobro poznavanje engleskog i maternjeg jezika uveliko olakšava razumijevanje logike programskih jezika, njihove sintakse i strukture. Predmet ima snažne veze s informatikom i bazama podataka, što omogućava učenicima da sveobuhvatno sagledaju rješavanje problema putem IKT-a.

B/ CILJEVI UČENJA I PODUČAVANJA PREDMETA

Na osnovu obrazovnih i društvenih zahtjeva savremenog doba te algoritmaskog načina razmišljanja i razumijevanja filozofije programiranja, ciljevi učenja i podučavanja predmeta Programiranje su:

1. Razvijanje sposobnosti analitičkog razmišljanja, logičkog povezivanja i razvoja programskih rješenja kroz algoritme, pseudokodove i dijagrame toka, koristeći strukture poput sekvence, selekcije i iteracije.
2. Razvijanje kreativnosti, inovativnosti i prilagodljivosti za ispravnu upotrebu raznih programskih jezika, s ciljem poticanja kreativnog promišljanja, eksperimentisanja i modifikacije rješenja kroz poznate tehnike razvoja softvera i različite programske paradigme.
3. Razumijevanje i praktična primjena različitih programskih struktura i struktura podataka kao što su sekvenca, selekcija, iteracija, nizovi, matrice, stringovi, datoteke, klase, objekti, metode i programiranje baza podataka.
4. Povezivanje i praktična primjena usvojenih znanja u konkretnim programskim rješenjima kroz strukturno i objektno orijentisano programiranje, uz akcenat na pravilnu primjenu sintakse, struktura i tipova podataka.
5. Primjena principa programiranja i projektne pismenosti kroz praktični rad, uključujući samostalnu ili timsku izradu programske dokumentacije, provjeru i zaštitu autorskih prava vlastitih rješenja, te etičko i odgovorno korištenje digitalnih tehnologija.

C/ OBLASNA STRUKTURA PREDMETNOG KURIKULUMA

I razred

1. Uvod u programiranje (pojam programiranja, kompajler, linker, faze programiranja)
2. Programski jezici (historijski razvopj programskih jezika, podjela i osobine generacija programskih jezika)
3. Algoritmi (definicija i svojstva algoritma, analiza problema, etape rješavanja problema, grafički zapis algoritma, algoritamske dtrukture, složeni algoritmi)
4. Struktura programa (upoznavanje sa strukturom programskog jezika (npr. C++) i strukturom programa, ključne riječi, identifikatori, definicija konstanti i promjenljivih, komentari)
5. Tipovi podataka (osnovni tipoivi podaaka u programskom jeziku, unos i prikaz podataka (cout, cin)
6. Izrazi inaredbe (operatori jezika C++, naredbe, izrazi, prvenstvo operatora, operator prdruživanja, aritmetički operatori, relacijski operatori, logičkioperatori, operatori inkrementiranja i dekrementiranja, naredbe pridruživanja, složene naredbe)
7. Tok programa (naredbe grananja (if, if – lse), naredbe višestrukog grananja (switch), naredbe ponavljanja (for, while, do – while), naredbe break i continue, naredba bezuslovnog skoka (go to)).

II razred

1. Nizovi (jednodimenzionalni i višedimenzionalni) (jednodimenzionalni nizovi, pristupanje elementima niza, pretraživanje i sortiranje niza, višedimenzionalni nizovi, pristupanje elementima matrice, ..)
2. Vektori (pretraživanje vektora, rotiranje vektora, invertovanje vektora, sortiranje vektora, sažimanje vektora, proširivanje vektora)
3. Funkcije (definisanje funkcije, parametri i argumenti funkcije, formalni parametri, lokalne i globalne promjenljive, rekurzivne funkcije)
4. Stringovi (definicija stringa, inicijalizacija stringa, pristup elementima stringa pomoću indeksa, upoznavanje sa osnovnim funkcijama za rad sa stringovima iz biblioteke)
5. Dinamičke strukture podataka (dinamičke varijable, pokazivači, jednostruko povezane liste, kružne liste, grafovi, struktura drveta, binarno stablo)
6. Datoteke (Otvaranje i zatvaranje datoteke, rad sa datotekama)

III razred

1. Osnovni koncepti objektno orjentisanog jezika (klasa, objekt, enkapsulacija, konstruktori, destruktori, nasljeđivanje, polimorfizam)
2. Objektni jezik i C jezik, šta je isto a šta različito (logički podaci, definisanje nabrojivog i strukturnog tipa, definisanje podataka na nivou bloka, preklapanje imena funkcije, reference, operator this, dinamički objekti)
3. Klase (Pojam klase, Enkapsulacija podataka, Razlika između klase i strukture, Podrazumijevani konstruktor, Konstruktor sa parametrima, Konstruktor kopije, Metode, Set metode, Get metode, Vaza između klasa)

4. Rukovanje izuzecima (rukovanje izuzecima, prijavljivanje izuzetaka, prihvatanje izuzetaka)
5. Bibliloteka komponenti (izrada projekta, forma, svojstva, metode, događaji, labela, dugme, događaji miša, zajednička svojstva za sve komponente, okvir za tekst (edit), panel, kastovanje komponente nad kojom se desio događaj, okvir za grupu, okvir za potvrdu, grupa radio dugmadi, događaj tastature, komponenta listbox, kombinovani okvir za tekst sa listom combobox, komponenta timer, dinamičko kreiranje komponenti, niz pokazivača na komponente, matrica pokazivača na komponente)

IV razred

1. Uvod u web tehnologije i osnove HTML-a i CSS-a (arhitektura web aplikacija (klijent–server), uloga HTML-a i CSS-a, struktura HTML dokumenta, elementi i atributi (naslovi, linkovi, liste, tabele, forme), CSS selektori, klase, ID, osnovno pozicioniranje)
2. Uvod u JavaScript i osnove klijentske logike (promjenljive, funkcije, događaji, rad sa DOM-om, validacija formi, povezivanje sa HTML-om, jednostavne interakcije korisnika sa stranicom)
3. Instalacija i konfiguracija PHP i razvojne okoline (instalacija lokalnog servera (XAMPP ili sl.), upoznavanje sa editorom (VS Code / Notepad++ / Visual Studio), prvi PHP program, ugradnja PHP-a u HTML)
4. Osnove PHP-a (sintaksa, promjenljive, izrazi, Uslovni izrazi i petlje, Funkcije i uključivanje fajlova, GET i POST metode, Rukovanje formama sa PHP-om)
5. Rad sa bazom podataka (MySQL) (Kreiranje baze i tabela, Povezivanje PHP-a sa MySQL-om, Unos, čitanje, izmjena i brisanje podataka (CRUD), Osnovne SQL naredbe (SELECT, INSERT, UPDATE, DELETE))
6. Praktičan razvoj dinamičke web stranice (Planiranje izgleda i strukture stranice, Korištenje HTML/CSS/JS + PHP/MySQL u jednom projektu, Validacija podataka, prikaz informacija, jednostavna sesija (login), Modularna struktura koda (header, footer, include))
7. Uvod u sigurnost web aplikacija (Osnovne sigurnosne prijetnje (XSS, SQL injection), Validacija i filtriranje korisničkog unosa, Zaštita formi i podataka, Pristup kontroli (login – osnovni koncept))
8. Izrada završnog mini projekta i prezentacija (Grupni ili individualni rad, Dokumentacija, Prezentacija, Evaluacija i povratna informacija)

Nastava predmeta Programiranje temelji se na razvijanju znanja i vještina koje omogućavaju učenicima da osmisle, kreiraju i analiziraju funkcionalne programske proizvode. Sadržaj predmeta strukturiran je kroz tri međusobno povezane oblasti:

A. Algoritmi
B. Reprezentacija i obrada podataka
C. Kontrola toka i komunikacija.

Svaka od ovih oblasti predstavlja ključni aspekt programiranja, a zajedno obuhvataju cjelovit proces rješavanja problema putem programskog jezika. Iako su tematski diferencirane, ove oblasti se u nastavi ne tretiraju strogo odvojeno. One se prirodno prožimaju i nadopunjuju

unutar različitih tematskih cjelina, u skladu s logikom izgradnje programskog rješenja. Bez obzira na programski jezik koji se koristi, učenici se podstiču da prepoznaju i primjenjuju ove koncepte kao integrisane cjeline.

A. Algoritmi

Oblast Algoritmi usmjerena je na razvoj sposobnosti učenika da probleme iz stvarnog svijeta sagledaju kroz prizmu računske logike, te da ih rješavaju konstruisanjem algoritama. To podrazumijeva identifikaciju problema, formulaciju strategije rješavanja i predstavljanje rješenja u obliku sekvence precizno definisanih koraka koje može interpretirati računar. Kroz ovu oblast učenici također analiziraju već postojeće algoritme, upoznaju se s njihovom strukturom i načinom rada, te razvijaju sposobnost poređenja njihove efikasnosti. Time se kod učenika jača algoritamski način razmišljanja i podstiče razumijevanje optimizacije u programiranju.

B. Reprezentacija i obrada podataka

U savremenom digitalnom okruženju, podaci predstavljaju osnovu svakog informacionog sistema. Oblast Reprezentacija i obrada podataka bavi se načinima kako podatke adekvatno predstaviti, pohraniti i efikasno obrađivati u digitalnom obliku. To uključuje rad s različitim tipovima podataka, strukturama podataka, kao i pristupima pohrani i pristupu informacijama. Učenici se kroz ovu oblast upoznaju s različitim modelima predstavljanja podataka, te izučavaju osnovne i napredne metode obrade, uključujući moderne pristupe poput objektno zasnovanog i objektno orijentisanog programiranja. Fokus je na razumijevanju kako struktura i organizacija podataka utiču na brzinu, pouzdanost i funkcionalnost programskog rješenja.

C. Kontrola toka i komunikacija

Oblast Kontrola toka i komunikacija obuhvata logički slijed izvršavanja naredbi unutar programa, kao i interakciju programa s korisnikom, drugim softverima i hardverom. Ključni aspekti ove oblasti odnose se na upravljanje tokom izvršavanja – uključujući grananje, petlje, funkcije i rukovanje izuzecima – što omogućava dinamičko i fleksibilno ponašanje programskog sistema. Pored kontrole toka, oblast uključuje i sve oblike komunikacije programa s okolinom. To podrazumijeva oblikovanje korisničkog interfejsa, upravljanje ulaznim i izlaznim operacijama, kao i interakciju s hardverskim komponentama i memorijskim resursima. Učenici uče kako se programi pisani na jezicima visokog nivoa prevode u strojni (mašinski) kod putem kompajlera i interpretera, te kako se ti programi pohranjuju i izvršavaju Ova oblast, osim tehničkog aspekta, podstiče i razvoj preciznosti, sistematičnosti, te kritičkog i analitičkog mišljenja. Doprinosi oblikovanju samostalnih, pouzdanih i komunikativnih budućih programera, osposobljenih za jasno izražavanje i efikasno rješavanje problema u digitalnom okruženju.

Oblasna struktura predmetnog kurikuluma Programiranja

U nastavku slijedi dio koji se odnosi na odgojno-obrazovne ishode koji su okosnica predmetnog kurikuluma i razrađeni su za svaku od tri oblasti (domene) na kojima se temelji. Odgojno-obrazovni ishodi pomažu nastavnicima u praćenju napretka učenika i u vrednovanju učeničkih postignuća. Tokom pripremanja procesa učenja i podučavanja nastavnik treba povezati odgojno-obrazovne ishode sa sadržajima navedenim u kurikulumu i metodama podučavanja. U tabelama su odgojno-obrazovni ishodi označeni šiframa. Skraćenice poput A.I.1. ili B.IV.3. označavaju redom: oblast kojoj ishod pripada (A. Algoritmi, B. Reprezentacija i obrada podataka i C. Kontrola toka i komunikacija), godinu podučavanja predmeta (I.-prvi razred, II.-razred, III. – treći razred i IV. – četvrti razred u IT gimnaziji), te redni broj odgojno-obrazovnog ishoda koji se podučava u sklopu navedene oblasti (1. – prvi ishod, 2. – drugi ishod, ...). Skraćenice TIT 3.1.1. ili npr. TIT 4.1.2. označavaju poveznice sa Zajedničkom jezgrom nastavnih planova i programa definisanih na ishodima učenja.

D/ ODGOJNO-OBRAZOVNI ISHODI

IT GIMNAZIJA

1. **razred IT gimnazije** /2 nastavna časa sedmično/70 nastavnih časova godišnje/

<table>
<tr><th colspan="2">Oblast: A/Algoritmi</th></tr>
<tr><th>Ishod učenja</th><th>Razrada ishoda</th></tr>
<tr><td>A.I.1. Analizira problem i razvija algoritamsko rješenje primjenom osnovnih algoritamskih struktura.</td><td>● Prepoznaje ključne elemente jednostavnog problema pogodnog za algoritamsko rješavanje.
● Razlaže problem na manje, logički povezane cjeline.
● Primjenjuje algoritamske strukture slijeda, grananja i ponavljanja pri izradi rješenja.
● Oblikuje jednostavan algoritam u tekstualnom ili grafičkom obliku (npr. pseudokod, dijagram toka).
● Objašnjava logiku i svrhu svake komponente izrađenog algoritma.
● Provjerava tačnost i efikasnost izrađenog algoritma kroz primjere.</td></tr>
<tr><td>Poveznice sa ZJNPP</td><td>TIT-4.1.1 TIT-4.1.2</td></tr>
<tr><th colspan="2">Ključni sadržaji</th></tr>
<tr><td colspan="2">Definicija i svojstva algoritama.
Analiza problema i algoritamski pristup rješavanju.
Etape algoritamskog rješavanja problema.
Osnovne algoritamske strukture: slijed, grananje, ponavljanje.
Pseudokod i dijagram toka kao oblici zapisa algoritma.
Testiranje i verifikacija jednostavnog algoritma.</td></tr>
<tr><th colspan="2">Preporuke za ostvarenje ishoda</th></tr>
<tr><td colspan="2">Kontekstualizovati zadatke: koristiti svakodnevne primjere iz života učenika (npr. algoritam za kuhanje čaja, odlazak u školu).
Kombinovati metode prikaza: podučavati i grafički (dijagrami toka) i tekstualni prikaz (pseudokod) algoritma.
Koristiti problemski orijentisanu nastavu: učenici rade u parovima ili grupama na analizi i razradi problema.
Podsticati refleksiju i diskusiju: učenici verbalizuju rješenja i razmjenjuju pristupe u razredu.
Korištenje softverskih alata: uključiti alate poput flowgorithm ili scratch.mit.edu za vizualizaciju algoritama.
Napredniji učenici: mogu se izazvati poređenjem dvaju algoritama za isti problem (po efikasnosti ili jasnoći).</td></tr>
<tr><td colspan="2"></td></tr>
<tr><td>A.I.2. Koristi grafičke i tekstualne metode za predstavljanje algoritama.</td><td>● Objašnjava osnovne simbole i pravila korištenja u dijagramima toka.
● Povezuje elemente dijagrama toka s odgovarajućim algoritamskim strukturama.
● Piše pseudokod koristeći osnovna pravila i strukture algoritma.
● Prebacuje algoritam iz jednog oblika zapisa u drugi (npr. iz dijagrama toka u pseudokod i obratno).
● Procjenjuje jasnoću i čitljivost algoritma u različitim oblicima zapisa.
● Provjerava tačnost i efikasnost izrađenog algoritma kroz primjere.</td></tr>
<tr><td>Poveznice sa ZJNPP</td><td>TIT-4.1.1 TIT-4.1.2</td></tr>
</table>

Ključni sadržaji
Oblici zapisa algoritama: dijagram toka, pseudokod. Elementi i simboli dijagrama toka (početak/kraj, obrada, uslov, tok). Pravila za pisanje pseudokoda. Korelacija između algoritamskih struktura i njihovog prikaza. Pretvaranje algoritma iz dijagrama u pseudokod i obratno.
Preporuke za ostvarenje ishoda
Prikazati dobre i loše primjere zapisa algoritama radi razvijanja osjećaja za jasnoću i standardizaciju. Koristiti radne listove sa zadacima konverzije algoritma iz teksta u dijagram i pseudokod. Uvesti standardne konvencije za pisanje pseudokoda (npr. korištenje ključnih riječi: BEGIN, END, IF, WHILE...). Rad u parovima: jedan učenik crta dijagram toka, drugi piše pseudokod za isti algoritam, zatim zamjenjuju uloge. Digitalni alati: koristiti alate poput Draw.io, Lucidchart ili Flowgorithm za izradu dijagrama toka. Povezivati s nastavnim jezikom (npr. C++): istaknuti kako se dijelovi pseudokoda mapiraju u konkretan kod.

A.I.3. Razlikuje osnovne etape algoritamskog rješavanja problema.	● Identifikuje osnovne faze u procesu algoritamskog rješavanja problema. ● Objašnjava svrhu svake etape (analiza problema, planiranje rješenja, dizajn algoritma, testiranje i evaluacija). ● Redoslijedno opisuje korake koje je potrebno preduzeti od problema do gotovog algoritma. ● Daje primjere za svaku fazu na konkretnim problemskim situacijama. ● Uočava posljedice preskakanja ili površne obrade pojedinih faza.
Poveznice sa ZJNPP	**TIT-4.1.1 TIT-4.1.2**

Ključni sadržaji
Faze algoritamskog pristupa rješavanju problema: • Analiza problema, • Planiranje rješenja, • Dizajniranje algoritma, • Testiranje i evaluacija algoritma. Uloga svake etape u procesu programiranja. Primjeri za svaku fazu na jednostavnim problemima.
Preporuke za ostvarenje ishoda
Koristiti problemski pristup: predstaviti učenicima jednostavan zadatak i zajedno s njima proći sve faze algoritamskog pristupa. Vizualni prikazi: napraviti dijagram koji prikazuje etape kao međusobno povezane korake u ciklusu razvoja algoritma. Rad u grupama: podijeliti učenike u grupe gdje svaka grupa fokusira jednu fazu i zatim zajednički sklapaju kompletno rješenje. Primjeri iz svakodnevnog života: npr. algoritam za odlazak u prodavnicu, spremanje doručka, itd. Refleksivne vježbe: zadati učenicima da napišu kratku analizu grešaka koje nastaju ako se neka faza preskoči. Uvezati s kasnijim fazama programiranja: istaći da se ove etape reflektuju i prilikom pisanja stvarnog koda.

A.I.4. Prepoznaje i primjenjuje poznata algoritamska rješenja u različitim problemskim situacijama. *(Napomena: ovaj ishod je sadržajno proširen u odnosu na prethodne tri tačke i temelji se na stavci "složeni algoritmi", koja implicira rad s poznatim algoritamskim obrascima.)*	● Prepoznaje zadatke koji se mogu riješiti primjenom poznatih algoritama (npr. zamjena vrijednosti, brojanje, pretraga, sortiranje). ● Opisuje način rada poznatih algoritamskih rješenja. ● Implementira algoritam koristeći odgovarajuće algoritamske strukture. ● Prilagođava poznata rješenja konkretnim situacijama kroz modifikaciju ulaznih i izlaznih podataka.

	● Analizira prednosti i ograničenja primijenjenog algoritma u odnosu na druge moguće pristupe.
Poveznice sa ZJNPP	**TIT-4.1.2 TIT-4.1.3 TIT-4.2.2**

Ključni sadržaji

Primjeri poznatih algoritama za:
- Razmjenu vrijednosti dvije promjenljive
- Zbrajanje i brojanje elemenata
- Pretraživanje niza (linearno)
- Jednostavno sortiranje (npr. selection sort, bubble sort).

Strukture koje podržavaju ove algoritme (petlje, uslovi, pomoćne promjenljive).
Analiza i poređenje efikasnosti jednostavnih algoritama.

Preporuke za ostvarenje ishoda

Zadaci sa šablonom: ponuditi učenicima zadatke koji se mogu riješiti poznatim algoritamskim obrascima.
Upoređivanje algoritama: dati dva algoritma za isti zadatak (npr. sortiranje) i diskutovati koji je efikasniji i zašto.
Zadaci u parovima: jedan učenik daje tekstualni opis problema, drugi bira i implementira poznati algoritam kao rješenje.
Upotreba pseudokoda i dijagrama: predstaviti algoritam u više oblika prije implementacije u kodu.
Refleksivne aktivnosti: učenici objašnjavaju zašto su izabrali određeni algoritam i kako su ga prilagodili zadatku.
Diferencirana nastava: naprednijim učenicima ponuditi složenije algoritme (npr. binarna pretraga) kao izazov.

Oblast: B/Reprezentacija i obrada podataka

Ishod učenja	Razrada ishoda
B.I.1. Prepoznaje i primjenjuje osnovne tipove podataka i pravila za definisanje konstanti i promjenljivih.	● Objašnjava razliku između konstanti i promjenljivih. ● Definiše konstante i promjenljive u skladu sa sintaksom programskog jezika (npr. C++). ● Navodi i prepoznaje osnovne tipove podataka (npr. int, float, char, bool). ● Povezuje tip podatka sa vrstom informacije koju predstavlja. ● Primjenjuje odgovarajuće tipove podataka u jednostavnim programskim zadacima.
Poveznice sa ZJNPP	**TIT-4.2.1**

Ključni sadržaji

Osnovni pojmovi: konstanta, promjenljiva.
Pravila imenovanja i deklarisanja promjenljivih i konstanti.
Tipovi podataka: cijeli brojevi (int), realni brojevi (float/double), karakteri (char), logičke vrijednosti (bool).
Asignacija i osnovne operacije s promjenljivima.

Preporuke za ostvarenje ishoda

Prikazati konkretne primjere: npr. const float PI = 3.14; i int broj = 5; uz objašnjenje svake komponente.
Kreirati zadatke za analizu: učenici dobijaju isječke koda i određuju tipove podataka i njihovu namjenu.
Rad u paru: jedan učenik zadaje vrstu informacije, drugi bira odgovarajući tip podatka i piše definiciju.
Povezati sa svakodnevnim kontekstima: npr. promjenljiva za broj godina, ocjena, temperatura itd.
Korištenje tablica i dijagrama: za poređenje karakteristika tipova podataka (veličina, domet, format).
Greške kao prilika za učenje: analizirati tipične greške kod definisanja promjenljivih (npr. pogrešno ime, neodgovarajući tip, redefinisanje).

B.I.2. Oblikuje izraze i koristi operatore za obradu podataka u skladu sa sintaksom programskog jezika.	● Navodi i prepoznaje osnovne operatore (aritmetičke, relacijske, logičke). ● Kombinuje operatore u jednostavne izraze u skladu s pravilima sintakse. ● Primjenjuje operatore pridruživanja, inkrementacije i dekrementacije.

	● Razlikuje prvenstvo operatora i koristi zagrade za kontrolu redoslijeda izvršavanja. ● Rješava jednostavne problemske zadatke korištenjem izraza i operatora.
Poveznice sa ZJNPP	**TIT-4.2.1**
Ključni sadržaji	
Operator pridruživanja (=). Aritmetički operatori (+, -, *, /, %). Relacijski operatori (==, !=, <, >, <=, >=). Logički operatori (&&, ||, !). Operator inkrementiranja i dekrementiranja (++ i --). Prvenstvo operatora i korištenje zagrada. Primjeri izraza u jeziku C++ i njihovo izvođenje.	
Preporuke za ostvarenje ishoda	
Radionice i mini testovi: kratki zadaci za rješavanje izraza uz obrazloženje redoslijeda izvršavanja. Vizualizacija operatora: tabela prvenstva operatora dostupna na plakatu ili u digitalnom obliku. Korištenje isječaka koda: učenici analiziraju izraze i određuju rezultat izvođenja. Paralelni rad: učenici najprije rješavaju izraze na papiru, zatim provjeravaju izvođenje u programskom jeziku. Diskusija o greškama: analizirati učestale greške u upotrebi operatora (npr. zamjena = sa ==). Zadaci za kreativno izražavanje: npr. zadati konkretan problem i tražiti od učenika da sami kreiraju izraz koji ga rješava.	
B.I.3. Demonstrira unos i prikaz podataka korištenjem osnovnih naredbi ulaza/izlaza.	● Objašnjava svrhu naredbi za unos (cin) i prikaz (cout) podataka u jeziku C++. ● Povezuje ulazne/izlazne naredbe sa odgovarajućim tipovima podataka. ● Piše jednostavne naredbe za unos vrijednosti u promjenljive. ● Prikazuje tekstualni i numerički izlaz na ekranu koristeći cout. ● Kombinuje unos i izlaz u okviru jednostavnih zadataka i programa. ● Koristi escape karaktere i manipulatore izlaza za formatiranje ispisa (\n, \t, setw, endl).
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**
Ključni sadržaji	
Naredbe cin i cout u C++. Osnovni operatori ulaza (>>) i izlaza (<<). Korištenje tekstualnih poruka uz unos/izlaz. Escape karakteri (\n, \t). Manipulatori izlaza (endl, setw). Kombinovanje više naredbi ulaza i izlaza. Jednostavni primjeri interakcije sa korisnikom.	
Preporuke za ostvarenje ishoda	
Zadaci sa simulacijom dijaloga: npr. unos imena, godina i ispis poruke dobrodošlice. Rad na konkretnim primjerima: zadaci poput unosa dvije vrijednosti i prikaza njihovog zbira. Naglasiti razliku između unosa i prikaza: posebno u vezi s redoslijedom naredbi i njihovim efektima. Korištenje vizualnih vodiča: dijagrami koji prikazuju tok podataka od tastature ka memoriji i od memorije ka ekranu. Formatiranje izlaza: pokazati različite efekte manipulatora izlaza na oblik prikaza. Samoprovjera: učenici analiziraju isječke koda i predviđaju što će biti prikazano na ekranu. Rad u paru: jedan učenik piše kod, drugi testira unos i ispis u IDE-u.	

Oblast: C/ Kontrola toka i komunikacija	
Ishod učenja	**Razrada ishoda**

C.I.1. Strukturira jednostavne programske zadatke primjenom naredbi za grananje i ponavljanje.	• Prepoznaje situacije u kojima se koristi grananje ili ponavljanje. • Primjenjuje naredbe if, if–else, switch za kontrolu toka programa na osnovu uslova. • Objašnjava razliku između različitih vrsta petlji (for, while, do–while) i primjenjuje ih u odgovarajućem kontekstu. • Upravlja brojačima i uslovima za ulazak/izlazak iz petlji. • Kombinuje strukture grananja i petlji u jednostavnom programskom zadatku. • Testira funkcionalnost programa i vrši ispravke u slučaju logičkih grešaka.
Poveznice sa ZJNPP	**TIT-3.2.2 TIT-4.2.1 TIT-4.2.2**
Ključni sadržaji	
Naredbe grananja: if, if–else, switch. Naredbe ponavljanja: for, while, do–while. Ključne riječi: break, continue, goto. Kontrola uslova i promjenljivih unutar petlje. Kombinacija grananja i petlji u rješavanju problema. Primjeri iz svakodnevnog života: kalkulator, brojanje, evaluacija unosa.	
Preporuke za ostvarenje ishoda	
Zadaci iz realnog konteksta: npr. provjera da li je broj paran/neparan, ispis brojeva od 1 do 10, meni opcija s switch naredbom. Poređenje različitih petlji: ista funkcionalnost implementirana kroz for, while i do–while. Vizualizacija toka: crtanje dijagrama toka za logiku grananja i petlji prije implementacije. Kod sa greškom: dati učenicima zadatke koji sadrže logičke greške u toku kontrole i tražiti analizu i ispravku. Razrada algoritma prije koda: insistirati na pisanju pseudokoda ili kratke verbalne analize prije programiranja. Zadaci sa rastućim nivoom složenosti: od jednostavnih do onih gdje se grananja ugniježđuju unutar petlji i obratno.	
C.I.2. Primjenjuje kontrolne naredbe i upravlja tokom izvršavanja programa.	• Objašnjava funkciju kontrolnih naredbi break, continue i goto u okviru petlji i grananja. • Prepoznaje situacije u kojima se koristi prekid i preskakanje dijela programskog toka. • Ispravno implementira kontrolne naredbe unutar struktura ponavljanja i grananja. • Analizira posljedice upotrebe kontrolnih naredbi na tok izvršavanja programa. • Identifikuje i ispravlja greške uzrokovane nepravilnim korištenjem kontrolnih naredbi.
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**
Ključni sadržaji	
Kontrolne naredbe: • break – prekid izvršavanja petlje ili switch strukture • continue – preskakanje trenutne iteracije • goto – bezuslovni skok (uz napomenu o preporučenoj umjerenoj upotrebi). Lokacija i efekat kontrolnih naredbi unutar različitih struktura. Uloga i ograničenja kontrolnih naredbi u strukturi programa. Primjeri iz prakse gdje kontrolne naredbe olakšavaju implementaciju.	
Preporuke za ostvarenje ishoda	
Kratki primjeri sa i bez kontrolnih naredbi: pokazati razliku u ponašanju programa. Razrada zadataka u dijelovima: prvo bez break/continue, zatim sa njihovim dodavanjem za optimizaciju toka. Zadaci sa uočenim greškama: učenici analiziraju kod gdje je break ili continue pogrešno primijenjen. Grafički prikaz toka: crtati dijagrame toka programa i uočiti gdje dolazi do prekida ili preskoka. Vođena diskusija: da li i kada koristiti goto ?	

Povezivanje s algoritamskom logikom: isticanje da je ispravno upravljanje tokom ključno za efikasnost i razumljivost programa.	
C.I.3. Objašnjava ulogu kompajlera, linkera i osnovnih faza u procesu razvoja programskog rješenja.	• Navodi osnovne faze razvoja programskog rješenja: analiza problema, pisanje koda, kompajliranje, povezivanje, testiranje i ispravljanje grešaka. • Objašnjava funkciju kompajlera u prevođenju izvornog koda u mašinski kod. • Opisuje ulogu linkera u povezivanju više programskih modula u jednu izvršnu cjelinu. • Povezuje greške prikazane tokom kompajliranja i linkanja s odgovarajućim fazama u razvoju programa. • Razlikuje sintaksne, semantičke i logičke greške u programiranju.
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**

Ključni sadržaji

Faze razvoja programskog rješenja:
- Analiza problema
- Pisanje izvornog koda
- Kompajliranje
- Linkanje
- Testiranje i otklanjanje grešaka.

Kompajler: funkcija i uloga u prevođenju koda.
Linker: povezivanje biblioteka i modula.
Tipovi grešaka u programiranju: sintaksne, semantičke i logičke.

Preporuke za ostvarenje ishoda

Prikaz procesa kroz dijagram: vizualizacija toka od pisanja koda do izvršnog programa.
Analiza grešaka iz prakse: pokazati greške koje nastaju tokom kompajliranja i linkanja, te ih zajedno analizirati.
Praktična demonstracija u IDE-u: učenici pišu jednostavan program i prate poruke kompajlera i linkera.
Razlikovanje grešaka: vježbe gdje učenici klasificiraju greške (npr. zarez umjesto tačka-zarez, neinicijalizovana promjenljiva).
Diskusija o važnosti uredne strukture koda: isticanje veze između jasnoće koda i lakšeg otklanjanja grešaka.
Povezivanje sa realnim procesom razvoja softvera: osvijestiti da kompajler i linker nisu samo alati, već važni dijelovi programerskog alata i tokova rada.

2. razred IT gimnazije /2 nastavna časa sedmično/70 nastavnih časova godišnje/

<table>
<tr><th colspan="2">Oblast: A/Algoritmi</th></tr>
<tr><th>Ishod učenja</th><th>Razrada ishoda</th></tr>
<tr><td>A.II.1. Primjenjuje algoritme za pretraživanje i sortiranje podataka u nizu i vektoru.</td><td>● Opisuje osnovne metode pretraživanja (linearna i binarna pretraga).
● Objašnjava principe rada algoritama sortiranja (npr. selection sort, bubble sort).
● Implementira algoritme pretraživanja i sortiranja u jeziku C++ koristeći nizove i vektore.
● Analizira ulazne i izlazne podatke u funkciji ispravnosti algoritma.
● Uporeduje efikasnost različitih algoritama na jednostavnim primjerima.</td></tr>
<tr><td>Poveznice sa ZJNPP</td><td>TIT-4.2.1 TIT-4.2.2</td></tr>
<tr><th colspan="2">Ključni sadržaji</th></tr>
<tr><td colspan="2">Jednodimenzionalni nizovi i vektori.
Linearna i binarna pretraga.
Algoritmi sortiranja: selection sort, bubble sort (osnovni koncepti).
Pristup elementima niza i vektora pomoću indeksa.
Poređenje složenosti algoritama (osnovni nivo).</td></tr>
<tr><th colspan="2">Preporuke za ostvarenje ishoda</th></tr>
<tr><td colspan="2">Koristiti konkretne primjere sa malim brojem elemenata: da bi učenici lakše pratili tok algoritma korak po korak.
Vizualizacija algoritama: ručno crtanje koraka sortiranja i pretrage na tabli ili korištenjem digitalnih animacija.
Razvijanje algoritma iz svakodnevnih konteksta: npr. pretraga imena u imeniku, sortiranje ocjena.
Rad u parovima: jedan učenik objašnjava postupak, drugi implementira.
Eksperimentisanje s različitim ulazima: kako bi učenici uočili razliku u ponašanju i efikasnosti algoritama.
Uvod u pojam složenosti algoritma kroz brojanje koraka, bez formalne notacije.</td></tr>
<tr><td colspan="2"></td></tr>
<tr><td>A.II.2. Analizira i primjenjuje rekurzivne algoritme u rješavanju problemskih zadataka.</td><td>● Objašnjava pojam rekurzije i razlikuje ga od iteracije.
● Identifikuje osnovne komponente rekurzivne funkcije: bazni slučaj i rekurzivni poziv.
● Analizira tok izvršavanja rekurzivne funkcije koristeći jednostavne primjere (npr. faktorijel, Fibonaccijev niz).
● Implementira jednostavne rekurzivne algoritme u programskom jeziku.
● Prepoznaje prednosti i ograničenja rekurzije u odnosu na iteraciju.</td></tr>
<tr><td>Poveznice sa ZJNPP</td><td>TIT-4.2.1 TIT-4.2.2</td></tr>
<tr><th colspan="2">Ključni sadržaji</th></tr>
<tr><td colspan="2">Definicija rekurzije.
Struktura rekurzivne funkcije: bazni slučaj, rekurzivni slučaj.
Primjeri: faktorijel, Fibonaccijev niz, stepenovanje broja.
Poređenje rekurzije i iteracije (osnovni principi).
Pregled načina poziva funkcija (stack pozivi, koncept pozadinskog toka).</td></tr>
<tr><th colspan="2">Preporuke za ostvarenje ishoda</th></tr>
<tr><td colspan="2">Korištenje dijagrama toka ili stabla poziva: za vizualno predstavljanje toka rekurzivnih poziva.
Rad „korak po korak“: analizirati kako se funkcija poziva i vraća kroz svaki nivo rekurzije.
Poređenje s iterativnim rješenjem: isti zadatak riješiti i rekurzivno i iterativno, pa diskutovati o razlikama.
Zadaci sa vođenom izgradnjom: učenici prvo napišu bazni slučaj, zatim proširuju funkciju.
Upozorenje na greške: istaknuti značaj baznog slučaja kako bi se izbjegla beskonačna rekurzija.
Korištenje debuggera u IDE-u: učenici prate tok izvršavanja rekurzivne funkcije u stvarnom vremenu.</td></tr>
</table>

A.II.3. Razlikuje korake u razvoju algoritma za rad s dinamičkim strukturama podataka.	● Objašnjava osnovni koncept dinamičkog zauzimanja memorije pomoću pokazivača. ● Identifikuje strukturu jednostruko povezane liste, kružne liste, binarnog stabla i grafova. ● Opisuje osnovne operacije nad dinamičkim strukturama: kreiranje, dodavanje, brisanje, pretraga. ● Navodi korake potrebne za oblikovanje algoritma koji koristi dinamičke strukture. ● Uočava prednosti i ograničenja dinamičkih struktura u odnosu na statičke (nizovi, vektori).
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**

Ključni sadržaji

Dinamičke promjenljive i rad s memorijom.
Pokazivači i njihova uloga u dinamičkoj alokaciji.
Struktura: čvor (node), veza (link), početni i završni pokazivač.
Jednostruko povezana lista, kružna lista, binarno stablo, graf.
Osnovne operacije: inicijalizacija, dodavanje, uklanjanje, pretraga elemenata.
Poređenje statičkih i dinamičkih struktura.

Preporuke za ostvarenje ishoda

Vizualizacija struktura podataka: crtati liste, stabla i grafove sa prikazima čvorova i veza.
Simulacija koraka algoritma: npr. dodavanje elementa u listu – učenici fizički povezuju „kartice“ s pokazivačima.
Korištenje pseudo-koda: radi postupnog razumijevanja logike bez fokusa na sintaksu.
Upotreba animiranih alata: kao što su Visualgo.net za demonstraciju struktura u pokretu.
Paralelno poređenje: prikazati isti problem riješen uz pomoć niza i uz pomoć liste, zatim diskutovati o razlici.
Rad u grupama: svaki tim razvija algoritam za jednu operaciju nad strukturom, zatim ga prezentira razredu..

Oblast: B/Reprezentacija i obrada podataka	
Ishod učenja	**Razrada ishoda**
B.II.1. Manipuliše elementima jednodimenzionalnih i višedimenzionalnih nizova.	● Deklariše i inicijalizuje jednodimenzionalne i višedimenzionalne nizove u jeziku C++. ● Pristupa pojedinačnim elementima niza pomoću indeksa. ● Upotrebljava petlje za unos, obradu i ispis elemenata niza i matrica. ● Realizuje operacije kao što su suma, prosjek, maksimum/minimum, transponovanje matrice. ● Primjenjuje jednostavne algoritme za pretraživanje i sortiranje elemenata niza.
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**

Ključni sadržaji

Jednodimenzionalni i dvodimenzionalni nizovi (matrice).
Deklaracija, inicijalizacija i pristup elementima.
Obrada nizova pomoću petlji (for, while).
Osnovne operacije nad nizovima (računanje sume, prosjeka, sortiranje, pretraga).
Obrada i prikaz matrica (unos, ispis, transponovanje).

Preporuke za ostvarenje ishoda

Zadaci sa realnim kontekstom: npr. analiza ocjena učenika (jednodimenzionalni niz), raspored sjedenja u učionici (matrica).
Grafički prikaz strukture: koristiti tabele da bi se lakše vizualizovao izgled niza i matrice.
Vođena implementacija u učionici: učenici korak po korak pišu kod uz objašnjenja nastavnika.
Upoređivanje petlji: rješavanje istog problema pomoću različitih struktura ponavljanja.
Greške kao alat za učenje: analizirati pogreške kod pristupa elementima izvan granica niza.

Mini-projekti: npr. unos i prikaz matrice, pronalazak elementa najveće vrijednosti, transponovanje kvadratne matrice.	
B.II.2. Koristi vektore i stringove za pohranu i obradu podataka.	● Deklariše i inicijalizuje vektore i stringove koristeći odgovarajuće biblioteke u jeziku C++ (<vector>, <string>). ● Pristupa elementima vektora i stringa pomoću indeksa. ● Primjenjuje osnovne metode i funkcije za obradu vektora (dodavanje, uklanjanje, sortiranje, rotiranje, sažimanje). ● Koristi funkcije iz biblioteke za rad sa stringovima (npr. length(), substr(), append(), find()). ● Kombinuje rad s vektorima i stringovima u jednostavnim programskim zadacima.
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**
Ključni sadržaji	
Vector i string klase u C++. Inicijalizacija i pristup elementima vektora i stringova. Operacije nad vektorima: dodavanje (push_back), uklanjanje (erase), sortiranje, rotiranje. Operacije nad stringovima: pristup znakovima, dužina (length()), izdvajanje podstringa (substr()), spajanje (append()), pretraga (find()). Petlje i funkcije za obradu podataka u vektorima i stringovima.	
Preporuke za ostvarenje ishoda	
Prikaz realnih primjera: npr. manipulacija listom imena (vektor stringova), analiza rečenice (rad sa znakovima). Korištenje biblioteka: uvesti praksu uključivanja <vector> i <string> uz objašnjenje njihove uloge. Zadaci za vježbu: sortiranje brojeva u vektoru, pronalaženje riječi u stringu, spajanje dva stringa. Vođene demonstracije: pokazati razliku između statičkih nizova i fleksibilnosti vektora. Rad u parovima: jedan učenik kreira strukturu podataka, drugi implementira operacije. Igre i kvizovi: npr. pravljenje programa za provjeru palindroma, brojanje riječi u rečenici, manipulacija karakterima.	
B.II.3. Implementira osnovne operacije nad dinamičkim strukturama podataka (pokazivači, liste, stabla).	• Objašnjava osnovne pojmove: pokazivač, čvor, veza, korijen stabla. • Deklariše pokazivače i koristi ih za dinamičko zauzimanje memorije. • Kreira i povezuje čvorove jednostruko povezane liste. • Primjenjuje osnovne operacije nad listama: dodavanje, uklanjanje i prikaz elemenata. • Kreira strukturu binarnog stabla i implementira osnovne operacije (npr. umetanje čvora, ispis preorder/inorder/postorder). • Objašnjava razliku između linearnih (liste) i hijerarhijskih (stabla) struktura.
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**
Ključni sadržaji	
Pokazivači i operatori * i &. Dinamička alokacija memorije: new, delete. Struktura čvora (npr. struct Node { int data; Node* next; };). Jednostruko povezane liste: kreiranje, dodavanje, brisanje, ispis. Osnove binarnog stabla: pojam korijena, lijevog i desnog podstabla. Operacije nad stablima: umetanje čvorova, pretraga, ispis u različitim redoslijedima.	
Preporuke za ostvarenje ishoda	
Korištenje vizualnih pomagala: crtanje struktura lista i stabala sa pokazivačima između čvorova.	

Postepena gradnja koda: od deklaracije strukture čvora do povezivanja i ispisa elemenata. Simulacija memorije: rad sa „papirnim“ čvorovima i pokazivačima u razredu. Rad u grupama: svaka grupa implementira jednu operaciju (dodavanje, ispis, brisanje), zatim ih integrišu. Zadaci s greškama: analiza i ispravljanje koda koji sadrži tipične greške sa pokazivačima. Korištenje online alata (npr. Visualgo.net): vizualizacija ponašanja struktura u realnom vremenu.

Oblast: C/ Kontrola toka i komunikacija	
Ishod učenja	**Razrada ishoda**
C.II.1. Organizuje strukturu programa korištenjem funkcija s parametrima i povratnim vrijednostima.	• Definiše funkciju s odgovarajućim povratnim tipom i listom parametara. • Razlikuje formalne i stvarne (realne) parametre funkcije. • Poziva funkciju s prosljeđivanjem vrijednosti po vrijednosti i po referenci. • Objašnjava razliku između lokalnih i globalnih promjenljivih u kontekstu funkcija. • Organizuje program u više funkcionalnih cjelina (funkcija) radi bolje modularnosti i čitljivosti. • Vraća rezultat iz funkcije korištenjem return izraza.
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**
Ključni sadržaji	
Definisanje funkcija: zaglavlje, tijelo funkcije, return vrijednosti. Parametri funkcija: formalni i stvarni, prosljeđivanje po vrijednosti i po referenci. Povratni tipovi: void, int, float, string. Modularizacija programskog koda. Opseg promjenljivih: lokalne i globalne promjenljive.	
Preporuke za ostvarenje ishoda	
Demonstracija modularnog pristupa: jedan veći zadatak podijeliti u manje funkcije (npr. unos, obrada, ispis). Praktični zadaci: izrada funkcije koja vraća maksimum iz dva broja, računanje prosjeka, validacija unosa itd. Poređenje načina prosljeđivanja: zadaci u kojima učenici posmatraju efekat promjene vrijednosti nad parametrom. Zadaci sa greškama: učenici otkrivaju greške poput neodgovarajućeg broja argumenata, nedostatka return vrijednosti itd. Pisanje funkcija bez korištenja main() na početku: fokus samo na razumijevanje i ispravno formiranje funkcija. Rad u parovima: jedan učenik piše funkciju, drugi je koristi u glavnom programu i testira.	
C.II.2. Razlikuje lokalne i globalne promjenljive u kontekstu funkcionalnog toka programa.	• Definiše pojmove lokalne i globalne promjenljive. • Objašnjava opseg važenja (scope) promjenljivih u okviru funkcija i izvan njih. • Identifikuje konflikte i greške koje nastaju usljed preklapanja imena promjenljivih. • Upoređuje ponašanje lokalnih i globalnih promjenljivih tokom izvršavanja programa. • Ispravno koristi lokalne i globalne promjenljive u skladu sa zahtjevima zadatka.
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**
Ključni sadržaji	
Opseg (scope) promjenljivih: lokalni i globalni kontekst. Pravila deklaracije i vidljivosti promjenljivih. Životni vijek promjenljive (lifetime). Sukobi imena i senčenje (shadowing). Primjeri interakcije između lokalnih i globalnih promjenljivih. Praktični primjeri u C++.	
Preporuke za ostvarenje ishoda	
Praktična demonstracija u IDE-u: učenici mijenjaju vrijednosti promjenljivih unutar i izvan funkcija. Korištenje dijagrama vidljivosti: vizualni prikaz hijerarhije i važenja promjenljivih u programu.	

Zadaci za uspoređivanje: isti program sa lokalnim i sa globalnim promjenljivim – uočiti razliku u ponašanju. Zadaci sa greškom: analizirati kod u kojem dolazi do konflikta između lokalne i globalne promjenljive. Grupna diskusija: prednosti i mane korištenja globalnih promjenljivih, koncept „dobre prakse“ u programiranju. Vođena refleksija: kada i zašto koristiti lokalne promjenljive radi sigurnosti i modularnosti koda.	
C.II.3. Primjenjuje osnovne postupke za rad s datotekama u cilju čuvanja i razmjene podataka.	• Objašnjava svrhu rada s datotekama u kontekstu trajnog čuvanja podataka. • Deklariše i koristi objekte tipa ofstream i ifstream iz biblioteke <fstream>. • Otvara i zatvara tekstualne datoteke za upis i čitanje podataka. • Upisuje i čita podatke iz datoteke korištenjem odgovarajućih naredbi (<<, >>, .open(), .close()). • Rješava jednostavne programske zadatke koji uključuju rad s datotekama (npr. čuvanje unosa, ispis sadržaja). • Uočava i rješava tipične greške prilikom rada s datotekama (npr. neuspješno otvaranje datoteke).
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**
Ključni sadržaji	
Uključivanje biblioteke <fstream>. Objekti ofstream (output file stream) i ifstream (input file stream). Metode .open(), .close(). Operatori za upis i čitanje (<<, >>). Provjera uspješnosti otvaranja datoteke (.is_open()). Osnovni primjeri rada s tekstualnim datotekama (upis i čitanje linija, brojeva, stringova).	
Preporuke za ostvarenje ishoda	
Realni zadaci: npr. unos i čuvanje liste imena u datoteku, ispis sadržaja datoteke u konzolu. Vođena demonstracija: korak-po-korak prikaz otvaranja datoteke, pisanja i zatvaranja. Upotreba IDE-a: uočavanje grešaka pri nepostojećim ili neotvorenim datotekama. Rad u parovima: jedan učenik piše funkciju za upis, drugi funkciju za čitanje, zatim ih testiraju zajedno. Diskusija o prednostima datoteka: zašto je važno sačuvati podatke i kada to radimo. Zadaci sa simulacijom datoteka: rad sa stvarnim .txt fajlovima u učionici – unos podataka i ručna provjera sadržaja.	

3. razred IT gimnazije /2 nastavna časa sedmično/70 nastavnih časova godišnje/

<table>
<tr><th colspan="2">Oblast: A/Algoritmi</th></tr>
<tr><th>Ishod učenja</th><th>Razrada ishoda</th></tr>
<tr><td>A.III.1. Analizira i primjenjuje osnovne koncepte objektno orijentisanog programiranja u dizajnu jednostavnih programskih rješenja.</td><td>● Objašnjava osnovne principe OOP-a: klasa, objekt, enkapsulacija, nasljeđivanje, polimorfizam.
● Identifikuje komponente klase: atribute, metode, konstruktore i destruktore.
● Povezuje koncept klase sa realnim objektima iz svakodnevnog života.
● Dizajnira i implementira jednostavne klase i objekte u C# jeziku.
● Primjenjuje konstruktore za inicijalizaciju objekata.
● Koristi enkapsulaciju za zaštitu podataka i pristupa im putem metoda (get/set).</td></tr>
<tr><td>Poveznice sa ZJNPP</td><td>TIT-4.2.1 TIT-4.2.2</td></tr>
<tr><th colspan="2">Ključni sadržaji</th></tr>
<tr><td colspan="2">Pojmovi: klasa, objekt, atribut, metoda.
Konstruktori (podrazumijevani, sa parametrima, kopije) i destruktori.
Enkapsulacija: zaštita podataka, pristup putem metoda.
Povezivanje stvarnih entiteta s klasama u programiranju.
Razlika između klase i strukture u C#.</td></tr>
<tr><th colspan="2">Preporuke za ostvarenje ishoda</th></tr>
<tr><td colspan="2">Povezivanje s primjerima iz svakodnevnog života: npr. klasa Automobil s atributima boja, snaga, broj vrata, metodom Pokreni().
Radionički pristup u Visual Studiu: učenici kreiraju jednostavne klase, instanciraju objekte i pozivaju metode.
Vođeni zadaci: korak-po-korak implementacija klase sa enkapsulacijom i jednostavnim ponašanjem.
Upotreba get i set metoda: umjesto direktnog pristupa atributima, uz objašnjenje sigurnosnih razloga.
Grupni rad: svaki član tima dizajnira klasu za jedan dio zajedničkog projekta (npr. igrice – Igrac, Protivnik, Igra).</td></tr>
<tr><td colspan="2"></td></tr>
<tr><td>A.III.2. Razvija algoritamsku logiku korištenjem nasljeđivanja, enkapsulacije i polimorfizma.</td><td>● Objašnjava pojam i svrhu nasljeđivanja u OOP-u.
● Kreira hijerarhiju klasa i primjenjuje koncept nadklase i podklase.
● Primjenjuje enkapsulaciju u baznoj i izvedenim klasama.
● Objašnjava koncept polimorfizma i koristi metode koje se ponašaju različito u različitim kontekstima.
● Koristi override i virtual ključne riječi u C# jeziku.
● Razvija algoritme u kojima se objekti različitih tipova obrađuju na zajednički način (polimorfizam u akciji).</td></tr>
<tr><td>Poveznice sa ZJNPP</td><td>TIT-4.2.1 TIT-4.2.2</td></tr>
<tr><th colspan="2">Ključni sadržaji</th></tr>
<tr><td colspan="2">Nasljeđivanje (: base, protected, override, virtual).
Enkapsulacija unutar hijerarhije klasa.
Polimorfizam: preklapanje metoda, pozivi preko referenci bazne klase.
Primjeri: klase Zivotinja → Pas, Macka s metodom Glas().
Praktične implikacije: proširivost i ponovna upotreba koda.</td></tr>
<tr><th colspan="2">Preporuke za ostvarenje ishoda</th></tr>
<tr><td colspan="2">Praktični primjer hijerarhije klasa: npr. Zaposleni → Profesor, Administrator sa metodom PrikaziDetalje().
Demonstracija preko zajedničke metode: više objekata poziva istu metodu, ali se ponašanje razlikuje u zavisnosti od klase.</td></tr>
</table>

Kreiranje funkcija koje primaju referencu bazne klase: radi prikaza snage polimorfizma. Vizualni prikaz hijerarhije: crtanje stabla nasljeđivanja, s prikazom preklopljenih metoda. Zadaci u kojima učenici samostalno dizajniraju jednostavnu OOP hijerarhiju. Kratki kvizovi i provjere znanja: fokus na prepoznavanje ispravne upotrebe override, virtual, base.	
A.III.3. Implementira mehanizme rukovanja izuzecima u funkciji stabilnosti i kontrole toka programa.	● Objašnjava pojam izuzetka (exception) i razliku između greške i izuzetka. ● Identifikuje situacije u kojima može doći do izuzetaka (npr. dijeljenje nulom, pristup nedostupnom resursu). ● Primjenjuje ključne riječi try, catch, finally u jeziku C# za obradu izuzetaka. ● Prijavljuje i prihvata izuzetke koristeći odgovarajuće klase (Exception, DivideByZeroException, itd.). ● Povezuje rukovanje izuzecima sa stabilnošću i pouzdanošću programa. ● Priprema program da se "graciozno" ponaša u slučaju nepredviđenih situacija.
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**
Ključni sadržaji	
Izuzetci (exceptions): definicija, primjer iz prakse. Blokovi try, catch, finally. Klase izuzetaka (System.Exception i izvedene klase). Kreiranje vlastitih izuzetaka (osnovni nivo). Uloga rukovanja izuzecima u profesionalnom softveru.	
Preporuke za ostvarenje ishoda	
Simulacija grešaka u kodu: npr. unos nule kao djelitelja, pokušaj otvaranja nepostojeće datoteke. Analiza poruka o izuzetku: učenici objašnjavaju značenje i porijeklo prikazane greške. Korištenje try–catch u svakodnevnim zadacima: npr. unos broja s tastature i obrada pogrešnog unosa. Primjeri dobrog i lošeg rukovanja izuzecima: učenici komentarišu kodove i uče iz pogrešaka. Diskusija o profesionalnom razvoju softvera: zašto su izuzeci ključni za korisničko iskustvo i stabilnost sistema. Vježbe sa više catch blokova: obrada različitih vrsta izuzetaka u istom programu.	

Oblast: B/Reprezentacija i obrada podataka	
Ishod učenja	**Razrada ishoda**
B.III.1. Modelira strukture podataka pomoću klasa i objekata u programskom jeziku C#.	● Definiše klase koje predstavljaju apstraktne i konkretne strukture podataka (npr. korisnik, proizvod, igra, figura). ● Uključuje atribute i metode relevantne za modeliranu strukturu. ● Primjenjuje različite vrste konstruktora za inicijalizaciju objekata. ● Koristi klase za stvaranje više instanci objekata i upravljanje njima u programskom rješenju. ● Povezuje klase u funkcionalne cjeline (npr. klasa Igra koristi klase Igrac, Tabla, Potez). ● Dokumentuje i komentariše strukturu podataka unutar koda.
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**
Ključni sadržaji	
Klasa kao model realnog entiteta. Atributi i metode kao elementi ponašanja objekta. Inicijalizacija objekata pomoću konstruktora. Organizacija međusobno povezanih klasa (sastavljanje i korištenje objekata).	

Primjeri: klase Knjiga, Korisnik, Narudžba, Igrač, Pozicija. Pregled razlike između modeliranja statičkih i dinamičkih odnosa.	
Preporuke za ostvarenje ishoda	
Primjeri iz svakodnevnog života: npr. klasa Film sa atributima naziv, trajanje, žanr i metodama PrikaziInfo(). Vođeni rad u Visual Studiu: učenici kreiraju jednostavne klase, testiraju ih u Main() metodi i diskutuju o strukturi. Korištenje UML dijagrama: učenici prvo nacrtaju klasni model prije implementacije. Zadaci sa više klasa: uvesti relacije između klasa (has-a, uses-a) kroz praktične projekte. Diskusija o apstrakciji: zašto nije potrebno da sve klase sadrže detalje implementacije, već samo opis ponašanja. Projektni rad: modeliranje jednostavnog sistema (biblioteka, igrica, naručivanje hrane).	
B.III.2. Koristi reference, nabrojive i strukturne tipove podataka u cilju optimizacije programskog koda.	● Objašnjava razliku između vrijednosnih i referentnih tipova podataka u jeziku C#. ● Primjenjuje referentne tipove (npr. klase, nizovi, objekti) u kreiranju fleksibilnog i efikasnog koda. ● Deklariše i koristi nabrojive tipove (enum) za ograničeni skup vrijednosti. ● Koristi strukturne tipove (struct) u kontekstu jednostavnih, manje kompleksnih objekata. ● Upoređuje upotrebu klasa i struktura u različitim scenarijima. ● Kombinuje reference i strukturne/nabrojive tipove u konkretnim zadacima.
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**
Ključni sadržaji	
Vrijednosni i referentni tipovi podataka u C#. Reference i ponašanje objekata kroz referencu. Nabrojivi tipovi (enum) – definicija, dodjela i upotreba. Strukturni tipovi (struct) – razlika u odnosu na klase. Scenariji korištenja enum i struct za čitljivost i organizaciju koda. Povezivanje s OOP konceptima.	
Preporuke za ostvarenje ishoda	
Vizuelni prikaz razlika: tabela ili dijagram koji pokazuje razliku između class, struct i enum. Rad na jednostavnim zadacima: npr. enum Boja { Crvena, Zelena, Plava } korišten u klasi Automobil. Zadaci sa referencama: prikazati kako promjena nad objektom putem jedne reference utiče na sve reference ka istom objektu. Diskusija u razredu: kada koristiti struct umjesto class – kriteriji za donošenje odluke. Kombinovanje koncepata: npr. struktura koja koristi enum, klasa koja koristi pokazivač na tu strukturu. Upareni rad: jedan učenik piše enum, drugi koristi vrijednosti u logici programa.	
B.III.3. Primjenjuje dinamičko kreiranje i upravljanje objektima u radu sa komponentama i podacima.	• Objašnjava koncept dinamičkog kreiranja objekata i komponenti u C#. • Kreira kontrole u toku izvršavanja programa koristeći operatore new i pokazivače na komponente. • Inicijalizuje, pozicionira i dodjeljuje svojstva komponentama dinamički (npr. dugmad, tekstualna polja, paneli). • Organizuje nizove i matrice pokazivača na komponente za sistematsko upravljanje većim brojem elemenata. • Briše dinamički kreirane objekte gdje je primjenjivo i vodi računa o pravilnom oslobađanju resursa. • Kombinuje logiku upravljanja podacima sa vizuelnim prikazom i interakcijom korisnika.

Poveznice sa ZJNPP	TIT-4.2.1 TIT-4.2.2

Ključni sadržaji

Dinamičko kreiranje objekata i komponenti (new, dodavanje kontrola u Form).
Pokazivači na komponente (npr. Button[], TextBox[,]).
Upravljanje svojstvima, događajima i rasporedom kontrola.
Nizovi i matrice pokazivača na kontrole.
Povezivanje sa logikom programa i podacima korisnika.
Primjeri: generisanje forme sa 10 dinamičkih dugmadi, kreiranje kviza, igrice itd..

Preporuke za ostvarenje ishoda

Zadaci sa postepenim proširivanjem: početi od jednog dugmeta, zatim ga pretvoriti u niz, pa u matricu.
Demonstracija preko igara ili animacija: dinamički kreirani elementi koji reaguju na događaje (npr. klikni me igre).
Koristiti petlje za kreiranje većeg broja komponenti: učenici uvide prednosti sistematične dinamike.
Zadaci sa povezivanjem na podatke: npr. dinamičko kreiranje polja na osnovu broja učenika iz datoteke.
Povezivanje s događajima: svaki dinamički objekat reaguje na klik, tastaturu ili tajmer.
Vođena diskusija: zašto i kada koristiti dinamičko kreiranje – fleksibilnost, skalabilnost, kontrola toka.

Oblast: C/ Kontrola toka i komunikacija

Ishod učenja	Razrada ishoda
C.III.1. Razvija grafički korisnički interfejs korištenjem komponenti iz biblioteke vizuelnih elemenata.	• Objašnjava osnovne pojmove GUI-a (grafički korisnički interfejs) i svrhu njegovog korištenja. • Identifikuje osnovne komponente GUI-a u C# (npr. Form, Label, Button, TextBox, Panel, CheckBox, RadioButton, ListBox, ComboBox). • Koristi razvojno okruženje (Visual Studio) za postavljanje i podešavanje vizuelnih elemenata na formi. • Podešava svojstva komponenti (tekst, boja, veličina, položaj, font). • Kreira osnovni izgled aplikacije kroz raspored komponenti i njihovo grupisanje (npr. korištenje GroupBox, Panel). • Primjenjuje događaje za osnovnu interakciju između korisnika i aplikacije (npr. Click, TextChanged).
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**

Ključni sadržaji

Grafički korisnički interfejs – osnovni pojmovi.
Osnovne GUI komponente: Label, TextBox, Button, ListBox, ComboBox, CheckBox, RadioButton, Panel, GroupBox.
Podešavanje svojstava komponente: Name, Text, Enabled, Visible, BackColor, ForeColor, Font, Size, Location.
Grupisanje komponenti i organizacija forme.
Raspoređivanje elemenata za čitljivost i upotrebljivost.

Preporuke za ostvarenje ishoda

Rad u Visual Studiu kroz demonstraciju i vježbu: učenici prate korake nastavnika, a zatim samostalno primjenjuju naučeno.
Zadaci za dizajn forme: npr. izraditi formular za prijavu korisnika, unos podataka, kviz ili mini-kalkulator.
Korištenje storyboard tehnike: učenici prvo nacrtaju izgled forme na papiru prije izrade u IDE-u.
Postepena izgradnja GUI-a: dodavanje jedne po jedne komponente uz objašnjenje funkcije i svojstava.
Podsticanje estetskog razmišljanja: diskusija o dizajnu i upotrebljivosti – jednostavnost, čitljivost, boje.
Uključiti kreativne zadatke: npr. uređivanje vlastite kontakt forme, aplikacije za evidenciju, igrice s GUI elementima.

C.III.2. Organizuje interakciju između korisnika i programa kroz događaje tastature, miša i tajmera.	• Objašnjava osnovne pojmove događaja i upravljanja događajima u C# (event-driven programming). • Registruje i implementira događaje miša (MouseClick, MouseMove, MouseDown) i tastature (KeyDown, KeyPress, KeyUp). • Koristi komponentu Timer za izvođenje akcija u pravilnim vremenskim intervalima. • Realizuje jednostavne aplikacije koje reaguju na korisničke akcije (npr. pomicanje objekta tastaturom, promjena boje na klik miša). • Kombinuje više događaja za postizanje kompleksnijeg interaktivnog ponašanja aplikacije. • Analizira ponašanje programa u zavisnosti od korisničkog unosa i vremenskih događaja.
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**
Ključni sadržaji	
Događaji u C# i princip obrade događaja. Osnovni događaji miša: Click, MouseMove, MouseDown, MouseUp. Osnovni događaji tastature: KeyDown, KeyPress, KeyUp. Komponenta Timer: svojstva (Interval, Enabled) i događaj Tick. Veza između GUI komponente i događaja (+= new EventHandler(...)). Primjeri: upravljanje objektom strelicama, reagovanje na klik, animacija pomoću Timer-a.	
Preporuke za ostvarenje ishoda	
Demonstracije u učionici: nastavnik pokazuje primjer reagovanja na događaj, učenici ga implementiraju i prilagođavaju. Zadaci s konkretnim ciljem: npr. napraviti program gdje se kvadrat pomjera strelicama, a tajmer ga automatski vraća u početni položaj. Igre i animacije kao motivacija: učenici izrađuju mini-igrice sa jednostavnim mehanizmima (klikni da pogodiš, reakcija na tastaturu). Praktične vježbe za povezivanje teorije s praksom: učenici testiraju efekte različitih događaja na istu komponentu. Vođena diskusija: kako događaji omogućavaju fleksibilno i responzivno ponašanje korisničkog interfejsa. Rad u paru: jedan učenik definiše izgled i osnovne komponente, drugi piše događaje za njih.	
C.III.3. Kreira jednostavne interaktivne programe i animacije korištenjem pokazivača na komponente i nizova.	• Definiše nizove i matrice pokazivača na komponente (Button[], PictureBox[,], itd.). • Inicijalizuje, pozicionira i dodjeljuje funkcionalnost komponentama kroz petlje. • Povezuje dinamički kreirane komponente s odgovarajućim događajima (npr. klik, tajmer). • Organizuje logiku programa u skladu sa strukturom komponenti i njihovim međusobnim odnosima. • Kombinuje elemente tajmera, tastature i događaja miša za realizaciju osnovne animacije ili interaktivne igre. • Testira i koriguje ponašanje komponenti u interakciji s korisnikom i tokom vremena.
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**
Ključni sadržaji	
Nizovi i matrice pokazivača na GUI komponente (Label, Button, PictureBox, TextBox, itd.). Dinamičko kreiranje i upravljanje komponentama u petlji. Dodjeljivanje događaja komponentama pomoću delegata. Upotreba Timer komponente za animaciju. Koordinacija logike i vizuelnog prikaza kroz nizove. Primjeri: igrica memorije, kviz, klikni me, zmija, vremenske reakcije.	
Preporuke za ostvarenje ishoda	

Projektni zadaci: izrada jednostavnih igara (npr. klikni na pravo dugme, kviz s više ponuđenih odgovora). Vođena implementacija: nastavnik pokazuje osnovu igrice, učenici dodaju nove funkcionalnosti i testiraju ponašanje. Petlje za raspored komponenti: npr. generisanje mreže dugmadi 4x4 za igru memorije. Događaji vezani uz indekse: učenici uče da identifikuju elemente niza kroz sender i castovanje. Povezivanje niza komponenti sa nizom podataka: logika i GUI se razvijaju paralelno. Diferencirani pristup: napredniji učenici mogu raditi sa animacijama i pokretnim objektima pomoću Timer-a.

4. razred IT gimnazije /2 nastavna časa sedmično/60 nastavnih časova godišnje/

<table>
<tr><td colspan="2">Oblast: A/Algoritmi</td></tr>
<tr><td>Ishod učenja</td><td>Razrada ishoda</td></tr>
<tr><td>A.IV.1. Primjenjuje skriptne programske jezike za obradu korisničkih zahtjeva i razvoj dinamičkih web funkcionalnosti.</td><td>● Objašnjava razliku između klijentskih (JavaScript) i serverskih (PHP) skriptnih jezika.
● Uključuje JavaScript i PHP kod u HTML stranicu i objašnjava njihovu ulogu u različitim fazama izvršavanja.
● Koristi PHP za obradu podataka sa forme (GET i POST metode).
● Koristi JavaScript za validaciju korisničkog unosa prije slanja na server.
● Kombinuje klijentsku i serversku logiku u jednostavnim zadacima (npr. registracija, prijava, kalkulator).
● Testira funkcionalnost aplikacije i prepoznaje gdje se izvršava koji dio koda (preglednik vs. server).</td></tr>
<tr><td>Poveznice sa ZJNPP</td><td>TIT-4.2.1 TIT-4.2.2</td></tr>
<tr><td colspan="2">Ključni sadržaji</td></tr>
<tr><td colspan="2">Osnove skriptnih jezika: JavaScript (klijent), PHP (server).
Umetanje JavaScript i PHP koda u HTML.
GET i POST metode.
Obrada formi i validacija podataka.
Interakcija korisnik–server: primjer rada web forme.
Zajedničko korištenje HTML, JavaScript i PHP-a.</td></tr>
<tr><td colspan="2">Preporuke za ostvarenje ishoda</td></tr>
<tr><td colspan="2">Početni zadaci sa jednostavnim formama: npr. unos imena i prikaz poruke u PHP-u, provjera praznog polja u JavaScript-u.
Uporedna demonstracija: pokazati isti zadatak s validacijom u JavaScript-u i u PHP-u – gdje, kada i zašto koristiti oba.
Rad u parovima: jedan učenik radi klijentsku validaciju, drugi serversku obradu, zatim zajedno integrišu rješenje.
Simulacija protoka podataka: učenici crtaju tok zahtjeva od korisnika do servera i nazad.
Testiranje na lokalnom serveru (XAMPP ili sl.): učenici odmah vide kako server odgovara na zahtjev.
Diskusija o prednostima i ograničenjima skriptnih jezika u web programiranju.</td></tr>
<tr><td colspan="2"></td></tr>
<tr><td>A.IV.2. Kreira funkcionalne dijelove aplikacije koristeći uslovne izraze, petlje i funkcije u JavaScript-u i PHP-u.</td><td>● Prepoznaje sintaksu uslovnih izraza i petlji u JavaScript-u i PHP-u.
● Primjenjuje grananje (if, else, switch) za upravljanje tokom izvršavanja u skriptnim jezicima.
● Koristi različite vrste petlji (for, while, foreach) za obradu podataka.
● Definiše i poziva vlastite funkcije s parametrima i povratnim vrijednostima.
● Razlikuje lokalni i globalni opseg promjenljivih unutar funkcija.
● Kombinuje funkcije, petlje i uslove u izgradnji modularnog i funkcionalnog dijela aplikacije.</td></tr>
<tr><td>Poveznice sa ZJNPP</td><td>TIT-4.2.1 TIT-4.2.2</td></tr>
<tr><td colspan="2">Ključni sadržaji</td></tr>
<tr><td colspan="2">Sintaksa i struktura uslovnih izraza i petlji u JavaScript-u i PHP-u.
Tipične funkcije: validacija, obrada unosa, formatiranje podataka.
Deklaracija i poziv funkcija.</td></tr>
</table>

Prosljeđivanje parametara i povratne vrijednosti. Opseg promjenljivih (var, let, const u JS; lokalne/globalne u PHP). Kombinovanje više struktura u jednoj funkciji.
Preporuke za ostvarenje ishoda
Zadaci s realnim primjerima: npr. provjera ispravnosti lozinke, brojanje brojeva u nizu, kreiranje tablice na osnovu korisničkog unosa. Pisanje paralelnih primjera u JS i PHP: isti zadatak (npr. zbir brojeva od 1 do n) prikazati u oba jezika i analizirati razlike. Rad u parovima ili grupama: jedna grupa implementira funkciju u JS, druga u PHP, zatim razmjena i analiza. Kreiranje jednostavnog "modula": funkcije koje se pozivaju iz forme (npr. kalkulator sa više operacija). Postavljanje pitanja "šta ako?": mijenjanje uslova i praćenje uticaja na tok programa. Upotreba vizualnih sredstava: dijagram toka koji prikazuje povezane funkcije i odluke unutar programa.

A.IV.3. Organizuje logiku programa u skladu s principima modularnosti i jasnoće server-klijent arhitekture.	● Objašnjava osnovne principe modularnog programiranja i razdvajanja funkcionalnosti. ● Organizuje HTML, CSS, JavaScript i PHP kod u odvojene datoteke i dijelove. ● Koristi include/require naredbe u PHP-u za dijeljenje zajedničkog koda (npr. header.php, footer.php). ● Razdvaja logiku klijenta (interakcije, validacije) i servera (obrada, čuvanje podataka). ● Razvija aplikaciju tako da je lako razumljiva, nadogradiva i ponovo upotrebljiva. ● Poštuje osnovne principe dobre strukture projekta (nazivi datoteka, direktorijumi, dokumentacija).
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**

Ključni sadržaji
Modularnost u web programiranju. Podjela između klijentskog i serverskog koda. Organizacija koda: odvojeni fajlovi za HTML, CSS, JS, PHP. Korištenje include, require, require_once. Struktura projekta: index.php, form.php, db.php, scripts.js, style.css, itd.. Praktični primjeri: login forma, kontakt forma, sistem za unos i prikaz podataka.
Preporuke za ostvarenje ishoda
Primjeri dobre prakse: pokazati primjer dobro strukturiranog i loše organizovanog projekta; uporediti čitljivost i fleksibilnost. Praktičan zadatak: učenici organizuju svoj web projekat koristeći jasno strukturirane fajlove i direktorije. Rad u timovima: podjela rada – jedan učenik zadužen za klijentsku logiku, drugi za serversku, treći za izgled. Vođena analiza: razlaganje postojećeg projekta na funkcionalne cjeline i prikaz njihove međusobne povezanosti. Upotreba skica ili mapa sajta: za planiranje funkcionalnosti i strukture projekta. Refleksivni zadaci: učenici opisuju u čemu im modularna organizacija pomaže i kako doprinosi razumljivosti koda.

Oblast: B/Reprezentacija i obrada podataka	
Ishod učenja	**Razrada ishoda**
B.IV.1. Struktuira i upravlja podacima u relacijskim bazama korištenjem SQL jezika i povezivanjem sa PHP-om.	● Objašnjava osnovne pojmove relacijskih baza podataka (tabela, red, kolona, primarni ključ). ● Kreira jednostavne baze i tabele u MySQL okruženju. ● Piše osnovne SQL naredbe za unos, čitanje, izmjenu i brisanje podataka (INSERT, SELECT, UPDATE, DELETE). ● Povezuje PHP skriptu sa bazom podataka pomoću mysqli ili PDO biblioteka.

	● Izvodi operacije nad bazom kroz PHP forme i prikazuje rezultate u web aplikaciji. ● Prepoznaje i rješava osnovne greške u povezivanju i radu sa bazom.
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**
Ključni sadržaji	
Relacijski model podataka. SQL naredbe: CREATE, INSERT, SELECT, UPDATE, DELETE. MySQL alati (npr. phpMyAdmin, MySQL shell). Povezivanje PHP-a sa bazom (mysqli_connect, query, fetch_assoc). Izvršavanje SQL naredbi iz PHP-a. Prikaz rezultata u HTML tabeli.	
Preporuke za ostvarenje ishoda	
Praktični rad uz phpMyAdmin: kreiranje baze, unos testnih podataka, vizuelno učenje strukture. Zadaci u parovima: jedan učenik piše SQL upite, drugi ih integrira u PHP aplikaciju. Vizualizacija relacija: crtanje ER dijagrama za jednostavne sisteme (npr. učenici i predmeti). Upotreba try–catch blokova za povezivanje: učenici testiraju spajanje i rješavaju greške. Povezivanje sa realnim primjerima: npr. sistem za rezervaciju, unos komentara, prijava korisnika. Diferencirani pristup: napredniji učenici koriste pripremljene upite i parametrizaciju za sigurnije rukovanje.	
B.IV.2. Koristi HTML forme i serverski kod za unos, validaciju, prikaz i ažuriranje podataka.	● Kreira HTML forme za unos različitih tipova podataka (tekst, brojevi, datumi, izbori). ● Povezuje forme sa PHP skriptama pomoću action i method atributa. ● Prikuplja podatke sa forme korištenjem $_GET i $_POST superglobalnih nizova. ● Primjenjuje osnovnu validaciju unosa pomoću JavaScript-a (klijent) i PHP-a (server). ● Upisuje podatke iz forme u bazu i prikazuje ih u HTML tabeli. ● Omogućava korisniku da ažurira i briše prethodno unesene podatke kroz web interfejs.
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**
Ključni sadržaji	
HTML elementi forme: <form>, <input>, <textarea>, <select>, <button>. Atributi: name, value, method, action. Prikupljanje podataka: $_POST, $_GET. Validacija unosa: JavaScript (osnovna), PHP (obavezna polja, tip podataka). Umetanje podataka u bazu i prikaz rezultata. Ažuriranje i brisanje podataka pomoću forme.	
Preporuke za ostvarenje ishoda	
Praktični zadaci: kreiranje forme za registraciju korisnika, unos komentara, rezervaciju termina. Upoređivanje GET i POST metoda: zadaci koji pokazuju razlike i preporučene primjene. Analiza sigurnosti: diskusija o osnovnim prijetnjama (npr. direktan unos SQL naredbi). Rad u fazama: kreirati formu → povezati s PHP-om → dodati validaciju → pohraniti u bazu → prikazati podatke. Zadaci sa greškom: učenici analiziraju i otklanjaju greške u povezivanju forme i servera. Učenje kroz refaktorisanje: unapređenje forme (npr. dodavanje select opcije, više tipova inputa).	
B.IV.3. Primjenjuje osnovne sigurnosne mehanizme za zaštitu podataka i korisničkog unosa.	• Objašnjava osnovne sigurnosne prijetnje u web aplikacijama (npr. SQL injection, XSS). • Prepoznaje ranjive tačke u aplikaciji povezane s korisničkim unosom i radom sa bazom. • Primjenjuje validaciju i filtriranje korisničkog unosa na strani klijenta i servera.

	• Koristi PHP funkcije za zaštitu od SQL injekcija (npr. mysqli_real_escape_string, prepared statements). • Uklanja HTML i JavaScript kod iz korisničkog unosa korištenjem htmlspecialchars, strip_tags i sličnih funkcija. • Demonstrira pravilno rukovanje osjetljivim podacima i pristupom (npr. login, session).
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**
Ključni sadržaji	
Osnovne prijetnje: SQL injection, XSS, krađa podataka. Validacija i sanitizacija podataka. PHP funkcije za zaštitu ulaza (htmlspecialchars, trim, strip_tags). SQL zaštita: korištenje prepared statements sa mysqli ili PDO. Razlika između klijentske i serverske zaštite. Primjeri sigurnog i nesigurnog koda.	
Preporuke za ostvarenje ishoda	
Prikaz "lošeg" koda: učenici otkrivaju i popravljaju ranjivosti u primjerima bez zaštite. Zadaci s kombinovanom validacijom: npr. JavaScript za osnovnu provjeru, PHP za konačnu verifikaciju. Primjena u postojećim formama: proširivanje prethodnih zadataka dodatnim slojem sigurnosti. Diskusija o stvarnim napadima: (primjereno uzrastu) – zašto sigurnost nije tehnička opcija, već nužnost. Grupni rad: timovi analiziraju aplikaciju i predlažu mjere zaštite za unesene podatke. Uvođenje termina poput "input sanitization" i "output escaping" kroz praktične primjere.	

Oblast: C/ Kontrola toka i komunikacija	
Ishod učenja	**Razrada ishoda**
C.IV.1. Razvija interaktivan korisnički interfejs korištenjem HTML-a, CSS-a i JavaScript-a.	• Strukturira HTML dokument koristeći semantičke elemente (header, nav, main, footer, itd.). • Primjenjuje CSS pravila za izgled i raspored elemenata (boje, margine, fontovi, pozicioniranje). • Povezuje CSS s HTML dokumentom putem <style> ili eksternih stilskih datoteka. • Koristi JavaScript za dodavanje dinamike (npr. promjena sadržaja, validacija forme, interakcija na klik). • Manipuliše DOM strukturom za izmjenu elemenata u realnom vremenu. • Kombinuje HTML, CSS i JavaScript u funkcionalnu i estetski prihvatljivu cjelinu.
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**
Ključni sadržaji	
Struktura HTML stranice (elementi, atributi, forme). Osnovni i napredni CSS selektori, pravila i pozicioniranje (display, flex, grid). Uvod u JavaScript: promjenljive, funkcije, događaji. DOM manipulacija (getElementById, innerHTML, addEventListener). Validacija formi i jednostavne vizualne reakcije. Primjeri: kalkulator, meni na klik, dinamički unos.	
Preporuke za ostvarenje ishoda	
Vođena izrada male aplikacije: npr. forma za unos + stilizacija + JavaScript za potvrdu. Korištenje browser developer alata (F12): učenici prate promjene DOM-a uživo. Primjeri iz svakodnevnog korištenja weba: analiziranje UI elemenata poznatih stranica (prijava, registracija, dropdown meni...). Zadaci u kojima učenici kombinuju tehnologije: svaka promjena u unosu reflektuje se kroz JavaScript i stilizira kroz CSS. Estetski izazovi: učenici dizajniraju „najljepši" kontakt formular koji uključuje osnovnu interakciju. Učenje kroz igru: npr. kviz sa dugmadima koja reaguju na klik.	

C.IV.2. Povezuje klijentsku i serversku logiku kroz HTTP zahtjeve i rukovanje podacima.	• Objašnjava osnovne koncepte HTTP protokola: zahtjev (request), odgovor (response), metode GET i POST. • Povezuje HTML formu sa PHP skriptom koja obrađuje podatke. • Koristi $_GET i $_POST za preuzimanje podataka sa klijentske strane. • Prikazuje rezultat obrade podataka iz PHP-a unutar HTML stranice. • Implementira jednostavnu dvosmjernu komunikaciju: unos → obrada → prikaz rezultata. • Analizira i rješava greške u komunikaciji između klijenta i servera.
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**

Ključni sadržaji

HTTP protokol: zahtjev i odgovor.
Metode GET i POST – razlike i upotreba.
Prijenos podataka između klijenta i servera.
PHP superglobali $_GET, $_POST, $_REQUEST.
Prikaz rezultata iz serverske obrade u HTML-u.
Osnovna obrada grešaka i validacija zahtjeva.

Preporuke za ostvarenje ishoda

Ilustracija toka podataka: grafički prikaz – korisnik → forma → server → baza → server → korisnik.
Praktičan zadatak: forma za izračunavanje vrijednosti (npr. kalkulator, konverzija valuta) uz serversku obradu.
Rad u parovima: jedan učenik kreira HTML formu, drugi PHP skriptu za obradu, zatim zamjene uloge.
Zadaci sa praćenjem protoka podataka: učenici koriste var_dump() i echo da razumiju gdje se šta izvršava.
Upotreba alata kao što su "Network" tab u browser developer alatima: učenje kroz vizualni prikaz HTTP zahtjeva.
Greške kao sredstvo učenja: učenici namjerno ostavljaju greške u method, name, action i otkrivaju posljedice.

C.IV.3. Kreira i prezentira funkcionalnu web aplikaciju integrisanjem različitih tehnologija.	• Planira strukturu aplikacije (funkcionalnosti, raspored stranica, povezanost elemenata). • Kombinuje HTML za strukturu, CSS za stilizaciju, JavaScript za interakciju i PHP za serversku logiku. • Povezuje aplikaciju sa MySQL bazom podataka za pohranu i prikaz podataka. • Primjenjuje osnovne principe sigurnosti u radu s korisničkim podacima. • Testira aplikaciju i otklanja funkcionalne i logičke greške. • Prezentira aplikaciju pred razredom uz objašnjenje korištenih tehnologija, logike rada i izazova tokom izrade.
Poveznice sa ZJNPP	**TIT-4.2.1 TIT-4.2.2**

Ključni sadržaji

Planiranje projekta i strukture direktorijuma.
Integracija HTML, CSS, JS i PHP koda.
Rad sa bazom podataka – dinamičko prikazivanje i unos podataka.
Sigurnosne mjere (validacija, sanitizacija, rad sa sesijama – osnovni nivo).
Testiranje i otklanjanje grešaka.
Dokumentacija i prezentacija projekta.

Preporuke za ostvarenje ishoda

Zadati okvirnu temu: npr. aplikacija za evidenciju knjiga, mini blog, anketa, kontakt sistem.
Rad u grupama ili samostalno: podjela na frontend i backend zadatke, planiranje i praćenje napretka.
Vođenje dnevnika rada: učenici bilježe šta su uradili, s kojim izazovima su se susreli i kako su ih riješili.

Korištenje predloženog obrasca za dokumentaciju: naslov, opis projekta, korištene tehnologije, opis funkcionalnosti. Formativna evaluacija: učenici prezentuju projekat kroz kratku demonstraciju i objašnjenje. Refleksija: učenici komentarišu šta bi unaprijedili, kako su sarađivali i koje su vještine razvili.

E/UČENJE I PODUČAVANJE

Učenje i podučavanje predmeta Programiranje u IT gimnaziji osmišljeno je kao dinamičan, savremen i učeniku prilagođen proces koji traje tokom sve četiri godine školovanja. Programiranje se realizuje kroz dva časa sedmično, a preporučuje se izvođenje nastave u blok-časovima kako bi se omogućio dublji rad na zadacima i projektnim aktivnostima. Nastava je usmjerena ka aktivnom učenju i razvoju digitalnih, analitičkih i kreativnih kompetencija koje učenicima omogućavaju da programiranje razumiju kao alat za rješavanje problema i izražavanje ideja.

U svakoj nastavnoj godini učenici postupno usvajaju znanja iz različitih područja programiranja, od osnovnih pojmova i algoritama, preko rada sa strukturama podataka i funkcijama, do objektno orijentisanog programiranja, skriptnih jezika i razvoja web aplikacija. Sadržaji su raspoređeni tako da učenici kroz postepeno složenije zadatke razvijaju algoritamski način razmišljanja, usvajaju logiku i strukturu programskih jezika, te uče kako se programski koncepti primjenjuju u stvarnim situacijama.

Preporučuje se izvođenje nastave u savremeno opremljenom informatičkom kabinetu. Poželjno je da svaki učenik ima pristup vlastitom računaru, dok u minimalnim uslovima najviše dva učenika mogu dijeliti jedan računar. Računari trebaju biti umreženi i imati pristup internetu, te instaliran odgovarajući softver - barem jedan jezik baziran na strukturnoj i/ili objektno-orijentisanoj paradigmi (npr. C++, Python, Java, C#, PHP i sl.), kao i odgovarajuća razvojna okruženja. Preporučuje se da raspored računara u učionici omogućava dobru preglednost i kretanje nastavnika među učenicima, npr. postavka u obliku potkovice, što olakšava interakciju, nadzor i podršku u radu.

Uloga nastavnika je da učenike potiče na istraživanje, samostalno učenje i timsku saradnju. Nastavnik podstiče učenike da analiziraju probleme, planiraju rješenja, pišu kod, testiraju i ispravljaju greške, vode računa o dokumentaciji, te se razvijaju u odgovorne i savjesne digitalne stvaraoce. Pri planiranju aktivnosti nastavnik uzima u obzir predznanja, stilove učenja i interesovanja učenika. Zadaci se oblikuju tako da promiču rješavanje problema, logičko razmišljanje, tehničku preciznost i kreativnu slobodu.

Poseban značaj imaju projektni zadaci, kroz koje učenici prolaze cijeli razvojni ciklus: od identifikacije problema i planiranja rješenja, preko implementacije i testiranja, do refleksije i prezentacije rezultata. Projekti mogu biti individualni ili timski, a uključuju izradu konkretnih programskih rješenja koja imaju svrhu i primjenu. Takav rad razvija kod učenika istraživački duh, osjećaj odgovornosti, sposobnost samoevaluacije i vještine komunikacije. Učenici vode evidenciju o svom radu, analiziraju uspjehe i poteškoće, dijele iskustva sa vršnjacima i uče iz međusobne interakcije.

Nastava se temelji na inkluzivnim principima gdje se uvažavaju individualne razlike, tempo rada prilagođava se mogućnostima učenika, a učenici koji izostaju ili imaju specifične potrebe uključuju se u rad kroz digitalne alate i platforme. Učionica programiranja treba biti prostor koji potiče saradnju, međusobno poštovanje, samostalnost, odgovornost i želju za učenjem.

Programiranje je multidisciplinarno i prirodno se povezuje s drugim nastavnim predmetima. Posebno je usklađeno s matematikom, tehničkom i informatičkom edukacijom, maternjim i stranim jezicima, ali i sa predmetima društvenih nauka kroz analizu korisničkih potreba, interfejs dizajn i komunikaciju sa softverom. Ovakav pristup učenicima pruža dublje razumijevanje tehnoloških procesa i omogućava im da razviju ne samo tehničke, već i životne vještine potrebne za uspjeh u savremenom digitalnom društvu.

F/VREDNOVANJE U PREDMETNOM KURIKULUMU

Vrednovanje u predmetu Programiranje ima ključnu ulogu u podršci učenicima tokom procesa učenja i napredovanja. Budući da ovaj predmet ne podrazumijeva usvajanje činjenica kroz memorisanje, već razvijanje logike, razumijevanje koncepata, praktičnu primjenu znanja i sposobnost rješavanja problema, vrednovanje mora biti kontinuirano, svrhovito i usmjereno na cjelokupni razvoj učeničkih kompetencija.

Učenici, osim što stiču teorijsko znanje, razvijaju i logičko, algoritamsko i kritičko mišljenje, sposobnost saradnje i rješavanja složenijih problema. Vrednovanje zato obuhvata različite pristupe i metode, usklađene s ciljevima učenja i razvojnim potrebama učenika. U predmetu Programiranje posebno je važno vrednovati kako krajnji rezultat, tako i proces rješavanja, uloženi trud, napredak i refleksiju učenika o vlastitom učenju.

Nastavnik prati učenike dok promišljaju, testiraju ideje, otklanjaju greške, sarađuju, dokumentuju rješenja i objašnjavaju svoj način razmišljanja. U tom smislu, uspjeh učenika se ne ogleda samo u ispravnosti koda, već i u njegovom pristupu, dosljednosti, kreativnosti i spremnosti da razumije zašto nešto (ne) radi. Posebnu pažnju treba posvetiti tome da se prepozna napredak, da se učenika vrednuje ne samo po tome gdje je stigao, već i koliko je od početne tačke uznapredovao.

Zato su zadaci u programiranju prilika za učenike da pokažu kako razmišljaju, a ne samo šta znaju. Svaka greška je dio učenja, i vrednovanje to mora uzeti u obzir. Umjesto ocjene koja samo konstatuje stanje, učeniku se daje prilika da svoje rješenje unaprijedi, da ga objasni, uporedi sa drugim pristupima i eventualno sam procijeni njegovu funkcionalnost i efikasnost.

Učenici se vrednuju kroz različite oblike aktivnosti, kao što su:

- **usmeno i pisano izražavanje razumijevanja pojmova i koncepata** (npr. objašnjenje algoritama, struktura podataka, logike koda),
- **praktični zadaci** na času ili u obliku domaćih zadataka (kodiranje, pronalaženje i otklanjanje grešaka, rad s nizovima, funkcijama, klasama, bazama podataka...),
- **projektni zadaci** i timski rad (izrada aplikacija, razvoj jednostavnih web rješenja, rješavanje problema iz stvarnog svijeta),
- **vođenje projektne dokumentacije i tehničkog dnevnika,**
- **samovrednovanje i vršnjačko vrednovanje,** gdje učenici razvijaju vještine objektivne procjene, kritičkog mišljenja i odgovornosti.

Ovakav pristup obuhvata tri domene učenja:

- **kognitivnu** - razumijevanje i tumačenje koncepata, izraza i struktura;
- **psihomotornu** - sposobnost praktične primjene znanja pri pisanju, testiranju i doradi koda;
- **afektivnu** - stavove učenika prema programiranju, motivaciju, saradnju i samostalnost.

Poseban značaj pridaje se funkcionalnom vrednovanju, gdje se ocjenjuje koliko učenik može samostalno osmisliti, implementirati i testirati programsko rješenje u realnim ili simuliranim situacijama, uz pravilnu primjenu standarda i dobre prakse programiranja.

U tom procesu važnu ulogu ima i učenička refleksija o sopstvenom radu, mogućnost da komentariše tuđa rješenja, da predloži poboljšanja i da prepozna dobre strane svog rada, što doprinosi vrednovanju koje razvija odgovornost i samostalnost. Nastavnik u tom procesu ne donosi zaključke sam, nego uključuje učenike u diskusiju o kvaliteti rješenja.

Ocjenjivanje se temelji na jasno postavljenim kriterijima koji su učenicima poznati i razumljivi. Ti kriteriji ne mjere samo ispravnost koda, već i njegovu čitljivost, logičku strukturu, primjerenost rješenju, dokumentaciju i korisničko iskustvo. Uzimaju se u obzir i elementi koji nisu odmah vidljivi, kao što je napor uložen u traženje rješenja, upornost u rješavanju izazova, prepoznavanje problema i predlaganje alternativnih pristupa.

Na kraju, vrednovanje u programiranju nije nešto što se dešava samo na kraju nastavne cjeline ili polugodišta. To je stalni dijalog između učenika i nastavnika, kroz koji se učeniku pomaže da razumije gdje se nalazi, kako napreduje i kako da svoje znanje dalje razvija.

G/PROFIL I STRUČNA SPREMA NASTAVNIKA

- Nastavu programiranja na osnovu člana 26. i 81. Zakona o srednjoj školi („Službene novine SBK“, broj 11/01 i 17/04) i Dopune nastavnih planova i programa za srednje škole broj: 01-38-953/09 mogu izvoditi lica sa završenim II (Drugim) ciklusom odgovarajućeg studija visokog obrazovanja (diplomski studij), sa akademskom titulom i stručnim zvanjem magistra za određenu obast kojim stiče 300 ECTS bodova.
- Nastavu predmeta Programiranje mogu izvoditi lica koja su završila odgovarajući fakultet na kome se stiče zvanje:
  - profesor informatike,
  - diplomirani informatičar,
  - diplomirani inženjer informatike,
  - profesor matematike i informatike ili drugog dvopredmetnog fakulteta u kojem je informatika ravnopravan predmet,
  - diplomirani inženjer elektrotehnike, smjer elektronika, smjer informatika ili računarstvo,
  - diplomirani inženjer informacijskih tehnologija,
  - Lica ostalih fakulteta koji obrazuju informatički kadar (VII/1; 300 ECTS), a odslušali su 4 semestra informatike (nastavni plan i program mora verifikovati Nastavno naučno vijeće na elektrotehničkom ili drugom srodnom tehničkom fakultetu).
- Ako osoba angažirana za izvođenje nastave u srednjoj školi tokom studija nije položila ispit iz pedagoško-psihološko-didaktičko-metodičke grupe predmeta, dužna je te ispite položiti u roku koji je utvrđen kantonalnim Zakonom o srednjem školstvu.
  Potrebnim pedagoško-psihološkim obrazovanjem nastavnika, stručnih saradnika i saradnika u srednjim školama smatra se pedagoško-psihološko obrazovanje koje obuhvaća obrazovna područja opće pedagogije, didaktike, metodike i psihologije odgoja i obrazovanja.(Dopune nastavnih planova i programa na bosanskome jeziku za srednje škole u SBK, broj:01-38-1525/12-2)
- Ukoliko lice u toku studija nije polagalo ispit iz pedagoško-psihološko-metodičke grupe predmeta, dužno je ove ispite položiti u roku od godine dana od dana stupanja na posao. (Dopuna nastavnih planova i programa za srednje škole, broj: 01-38-953/09)
- Profesori koji nisu navedeni u profilima, a stekli su pravo na izvođenje nastave po prethodnim propisima, zadržavaju stečeno pravo pod uvjetom da su zaposleni na neodređeno vrijeme. (Izmjena i dopuna nastavnih planova i programa za devetogodišnje osnovne škole koje nastavu realiziraju na bosanskom jeziku u Srednjobosanskom kantonu, broj: 01-34-76/2021)